An Analytic Tableaux based Characterization of Data Base Repairs for Consistent Query Answering*

(Preliminary Report)

Leopoldo Bertossi† and **Camilla Schwind**‡

Abstract

In this article, we characterize in terms of analytic tableaux the repairs of inconsistent relational data bases, that is data bases that do not satisfy a given set of integrity constraints. For this purpose we provide closing and opening criteria for branches in tableaux that are built for data base instances and their integrity constraints. We use the tableaux based characterization as a basis for algorithms for consistent query answering, that is for retrieving from the data base answers to queries that are consistent wrt the integrity constraints.

1 Introduction

Analytic tableaux [2] are used as formal deductive systems for propositional and predicate logic. Similar in spirit to resolution, but with some important methodological and practical differences [4], they are mainly used for producing formal refutations from a contradictory set of formulas. The set of formulas is inconsistent whenever all the branches in the tableau can be closed.

In this paper we extend the tableaux methodology to deal with a relational data base instance plus a set of integrity constraints that the first fails to satisfy, and in consequence, both inputs together can be considered as building an inconsistent set of sentences. We show how to close branches in a tableau in such a situation.

In [1], the notion of consistent answer to a query posed to an inconsistent data base was defined: a tuple is a consistent answer if it is an answer, in the usual sense, in every possible repair of the inconsistent data base. A repair is a new data base instance that satisfies the integrity constraints and differs from the original instance by a minimal set of changes wrt set inclusion.

In [1, 3], also a computational methodology was presented to obtain such consistent answers. Nevertheless, it has some limitations in terms of the syntactical form of integrity constraints and queries it can handle. In particular, it does not cover the case of existential queries and constraints.

The technique of "opening tableaux" was introduced in [6] for a solution to the frame problem, and in [8, 9] for applying tableaux methods to default logic.

*Work supported by FONDECYT Grant # 1000593; and ECOS/CONICYT Grant C97E05.

†P. Universidad Catolica de Chile, Escuela de Ingenieria, Depto. Ciencia de Computacion, Casilla 306, Santiago 22, Chile. bertossi@ing.puc.cl

‡Contact author. LIM, Faculté des Sciences de Luminy, 163 Avenue de Luminy, Case 901, 13288 Marseille, Cedex 9, France. schwind@lim.univ-mrs.fr

In this paper we show how to open tableaux for data base instances plus their constraints, and this notion of opening is applied to characterize and represent by means of a tree structure all the repairs of the original data base. Finally, we sketch how this representation could be used to retrieve consistent query answers. At least at the theoretical level, the methodology introduced in this paper could be applied to any kind of first order queries and constraints.

This paper presents ongoing work and some -we think- promising ideas and methodologies that deserve and require further investigation.

2 Inconsistent Data Bases and Repairs

In this paper a data base instance is given by a finite set of finite relations on a data base schema. A data base schema can be represented in logic by a typed first-order language, $\mathcal{L}$, containing a finite set of sorted predicates and a fixed infinite set of constants D. The language contains a predicate for each data base relation and the constants in D correspond to the elements in the data base domain, that we will also denote by D. That is every data base instance has an infinite domain D. We also have a set of integrity constraints IC expressed in language $\mathcal{L}$. These are first-order formulas which the data base instances are expected to satisfy. In spite of this, there are realistic situations where a data base may not satisfy its integrity constraints [1]. If a data base instance satisfies IC, we say that it is consistent (wrt IC), otherwise we say it is inconsistent. In any case, we will assume from now on that IC is a consistent set of first order sentences.

A data base instance r can be represented by a finite set of ground atoms in the data base language, or alternatively, as a Herbrand structure on this language. In consequence, we can say that a data base instance is consistent, wrt IC, when its corresponding Herbrand structure is a model of IC.

The active domain of a data base instance r is the set of those elements of D that explicitly appear in r. The active domain is always finite and we denote it by $Act(r)$. We may also have a set of built-in (or evaluable) predicates, like equality, arithmetical relations, etc. In this case, we have the language $\mathcal{L}$ possibly extended with these predicates. In all data base instances each of these predicates has a fixed and possibly infinite extension. Of course, since we defined data base instances as finite sets of ground atoms, we are not considering these built-in atoms as members of data base instances.

An inconsistent data base[1] has "most" of its data content still consistent wrt IC and can still provide "consistent answers" to queries posed to them. This notion was defined and analyzed in [1]. This was done on the basis of considering all possible changes to r, in such a way that it becomes a consistent data base instance. Then the query is answered with respect to all possible repaired data base instances.

There may be several ways to repair the data base. Of course, changes should be minimal and just sufficient to obtain a consistent data base instance. Next, we present a definition of "change", that is based on a notion proposed by Winslett [11] for knowledge base updates and was also used in [1]. This definition is based on a notion of minimal distance between models using symmetric set difference Δ.

[1] Sometimes we will simply say "data base" instead of "data base instance".

Definition 1 *Given data bases instances*[2] r, r' *and* r'', *we say that* r' *is closer than* r'' *to* r, *iff* $r \Delta r' \subseteq r \Delta r''$. *This is denoted by* $r' \leq_r r''$.

Only data base predicates are taken into account for the notion of distance. In particular, we do not change built-in predicates since built-ins have a fixed extension. We use this notion of distance for defining "repairs" of inconsistent data base instances. The idea is to choose consistent data bases which are as close as possible to the inconsistent data base instance.

Definition 2 *[1] (a) Given data base instances* r *and* r', r' *is a* repair *of* r, *if* $r' \models IC$ *and* r' *is a minimal element in the set of instances wrt the order* $\leq_r$.
(b) Given a data base instance r, *a set* IC *and a first order query* $Q(\bar{x})$, *we say that a ground tuple* $\bar{t}$ *is a consistent answer to* Q *in* r *wrt* IC *iff* $r' \models Q[\bar{t}]$ *for every repair* r' *of* r *(wrt* IC*).*

Example 1 *Consider* $IC = \forall x, y, z(Supply(x,y,z) \wedge Class(z,T_4) \rightarrow x = C)$ *and the inconsistent data base* $r = \{Supply(C, D_1, I_1), Supply(D, D_2, I_2), Class(I_1, T_4), Class(I_2, T_4)\}$. *Here we have two possible (minimal) repairs, one obtained by deleting* $Supply(D, D_2, I_2)$ *from* r *and the other one, by deleting* $Class(I_2, T_4)$ *from* r.

3 Analytic Tableaux and Data Base Repairs

We will use analytic tableaux to represent data base repairs and characterize consistent query answers. For this we will need a special form of tableaux, suitable for representing data base instances and their integrity constraints.

Given a data base instance r and a finite set of integrity constraints IC, we first compute the tableau, $TP(IC \cup r)$, for IC and r. This tableau has as root node the set of formulas $IC \cup r$. This tableau should be closed, that is the tableau has only closed branches, if and only if r is inconsistent. By suppressing literals in every closed branch we can transform r into a consistent data base instance and thus obtain a repair of the data base. For all this to work, we must take into account, when computing the tableau, that r represents a data base instance and not just a set of formulas, in particular, that the absence of positive information means negative information, etc. (see section 3.2). Next, we give a brief review of first order analytic tableaux.

3.1 Analytic tableaux

The analytic tableaux method was first introduced by Beth [2], and is best described in Smullyan [10] and Fitting [5]. It is a syntactically oriented refutation system.

The tableau of a set of formulas is obtained by recursively *breaking down* the formulas into subformulas, obtaining sets of sets of formulas. These are the usual Smullyan's classes of formulas:

[2] We are assuming here and everywhere in the paper that all data base instances have the same predicates and domain.

α	α_1	α_2	β	β_1	β_2
$f \wedge g$	f	g	$f \vee g$	f	g
$\neg(f \vee g)$	$\neg f$	$\neg g$	$\neg(f \wedge g)$	$\neg f$	$\neg g$
$\neg(f \rightarrow g)$	f	$\neg g$	$f \rightarrow g$	$\neg f$	g

γ	$\gamma(p), p$ any parameter	δ	$\delta(p), p$ Skolem parameter
$(\forall x) f$	$f[x/p]$	$(\exists x) f$	$f[x/p]$
$\neg(\exists x) f$	$\neg f[x/p]$	$\neg(\forall x) f$	$\neg f[x/p]$

A tableaux prover produces a formula tree. An α-rule adds new formulas to branches, a β-rule splits the tableau and adds a new branch. Given a formula φ, we denote by $TP(\varphi)$ the tree produced by the tableaux system. We can think of this tree as the set of its branches.

Notice that the original set of constants in the language, in our case, D, is extended with a set of new constants, P, the so-called Skolem functions or parameters. These parameters, that we will denote by $p, p_1, \ldots$, have to be new at the point of their introduction in the tree in the sense that they have not appeared so far in the (same branch of the) tableau. When applying the γ-rule, the parameter can be any of the old or new constants.

A tableau branch is *closed* if it contains a formula and its opposite, otherwise it is *open*. Every open branch corresponds to a model of the formula. If $B \in TP(\varphi)$ and B is open, then the set of ground atoms on B is a model of φ. The completeness theorem for tableaux theorem proving [10] states that: F is a theorem iff $TP(\{\neg F\})$ is closed.

We consider TP not only as a theorem prover (or consistency checker) for formulae but also as an application from (sets of) formulas to trees which has some useful properties. Thus, operations on tableaux can be defined on the basis of the logical connectives occurring inside the formulas involved (see lemma 3 in the appendix).

3.2 Tableaux and data base instances

In data base theory, we usually make some assumptions [3]: (a) Unique Names Assumption (UNA): If a and b are different constants, then $a \neq b$ holds in r. (b) Closed World Assumption (CWA): If r is a data base instance, then for any ground data base atom $P(c)$, if $P(c) \notin r$, then $\neg P(c)$ holds for r, more precisely, implicitly, $\neg P(c)$ belongs to r.

When computing a tableau for a data base instance r, we do not add explicitly the formulas corresponding to the UNA and CWA, rather we keep them implicit, but taking them into account when computing the tableau. This means, for example, that the presence on a tableau branch of a formula $a = b$, for different constants a, b in D, closes the branch.

Given a data base r and integrity constraints IC, we will generate the tableau $TP(IC \cup r)$. Every branch B of this tableau will be of the form $I \cup r$, where $I \in TP(IC)$ (see lemma 3).

$TP(IC \cup r)$ is defined as in section 3.1, but we still have to define the closure conditions for tableaux associated to data base instances. Before we present some motivating examples.

[3] Actually, it is possible to make all these assumptions explicit and transform the data base instance into a first-order data base theory [7].

Example 2 *(example 1 continued) In this case, $TP(IC \cup r)$ is the following tree*

$$\forall x, y, z(Supply(x, y, z) \ \wedge \ Class(z, T_4) \ \rightarrow \ x = C)$$
$$Supply(D, D_2, I_2)$$
$$Supply(C, D_1, I_1)$$
$$Class(I_1, T_4)$$
$$Class(I_2, T_4)$$
$$|$$
$$Supply(C, D_1, I_1) \ \wedge \ Class(I_1, T_4) \ \rightarrow \ C = C$$
$$|$$
$$Supply(D, D_2, I_2) \ \wedge \ Class(I_2, T_4) \ \rightarrow \ D = C$$

$$\neg Supply(C, D_1, I_1) \qquad \neg Class(I_1, T_4) \qquad C = C$$
$$\dots \qquad \dots$$
$$\times \qquad \times$$

$$\neg Supply(D, D_2, I_2) \qquad \neg Class(I_2, T_4) \qquad D = C$$
$$\times \qquad \times \qquad \times$$

The last branch is closed because $D = C$ is false in the data base (alternatively, because $D \neq C$ is implicitly in the data base). We can see that $TP(IC \cup r)$ is closed. r is inconsistent wrt IC. The nodes $Supply(C, D_1, I_1) \wedge Class(I_1, T_4) \rightarrow C = C$ and $Supply(D, D_2, I_2) \ \wedge \ Class(I_2, T_4) \ \rightarrow \ D = C$ are obtained by applying the γ-rule to $\forall x, y, z(Supply(x, y, z) \ \wedge \ Class(z, T_4) \ \rightarrow \ x = C)$. Application of the β-rule to $Supply(D, D_2, I_2) \ \wedge \ Class(I_2, T_4) \ \rightarrow \ D = C$ produces the same subtree for all three leaves ($\neg Supply(C, D_1, I_1)$, $\neg Class(I_1, T_4)$ and $C = C$). In the above figure, we indicate this subtree by "$\dots$". Here we omit the explicit addition of the subtree to the first two branches, because these branches are already closed.

In tableaux with equality, we also need extra rules for transitivity of equality and for replacing a term t in a predicate P by t' whenever $t = t'$ belongs to the same tableau branch (see example 10 in the appendix). In consequence, we will assume that we have a tableaux prover with reflexivity and paramodulation [5]. It will be simpler to define the closure rules for data base tableaux, if we skolemize existential formulas before developing the tableau ([4]). We assume from now on that all integrity constraints are skolemized by means of a set of Skolem constants and function symbols $P \cup F$.

Example 3 *Consider the referential IC : $\forall x \ (P(x) \rightarrow \exists y \ Q(x, y))$, and the inconsistent data base instance $r = \{P(a), Q(b, d)\}$, for $a, b, c \in D$. With an initial skolemization, we can develop the following tableau $TP(IC \cup r)$. In it, the second branch closes because $Q(a, f(a))$ does not belong to the data base instance.*

$$
\begin{array}{c}
\forall x\ (P(x) \rightarrow Q(x, f(x))) \\
P(a), Q(b, d) \\
| \\
P(a) \rightarrow Q(a, f(a)) \\
\begin{array}{cc}
/ & \backslash \\
\neg P(a) & Q(a, f(a)) \\
\times & \times
\end{array}
\end{array}
$$

Example 4 *Consider the inconsistent data base* $r_1 = \{Q(a), Q(b)\}$ *wrt the* $IC:\ \exists x\ P(x)$*. After having skolemized* $\exists x\ P(x)$ *into* $P(p)$*, a tableau proof for the inconsistency is the following,*

$$
\begin{array}{c}
P(p) \\
Q(a), Q(b) \\
\times
\end{array}
$$

The only branch closes because $P(p)$ *does not belong to the data base.*

Let us now change the data base instance to $r_2 = \{P(a), P(b)\}$*, keeping the integrity constraint. Now, the data base is consistent. Nevertheless, if we produce the following tableau* $TP(IC \cup r_2)$*:*

$$
\begin{array}{c}
P(p) \\
P(a), P(b)
\end{array}
$$

This time we do not want the tableau to close, and thus sanctioning the inconsistency of the data base. The reason is that we could make p take any of the values in the active domain $\{a, b\} \subseteq D$ *of the data base.*

A similar situation can be found in example 3. If we change the data base instance to $\{P(a), Q(a, d)\}$*, which is consistent, we obtain*

$$
\begin{array}{c}
\forall x\ (P(x) \rightarrow Q(x, f(x))) \\
P(a), Q(a, d) \\
| \\
P(a) \rightarrow Q(a, f(a)) \\
\begin{array}{cc}
/ & \backslash \\
\neg P(a) & Q(a, f(a)) \\
\times &
\end{array}
\end{array}
$$

Now we do not close the second branch because we may define a f *as a function from the active domain into itself that makes* $Q(a, f(a))$ *become a member of the data base, actually by defining* $f(a) := d$*.*

Example 5 *Consider* $IC:\ \exists x\ \neg P(x)$ *and the consistent data base instance* $r = \{P(a)\}$*. The tableau* $TP(IC \cup r)$ *after skolemization of* IC *is:*

$$
\begin{array}{c}
\neg P(p) \\
P(a)
\end{array}
$$

We do not want this tableau to close. Actually, this tableau cannot be closed, because p must be a new parameter, not occurring in the same branch of the tableau and it is not the case that $P(p) \in r$ (alternatively, we may think of p as a constant that can be defined as any element in $D \setminus \{a\}$, that is in the complement of the active domain of the data base.

Definition 3 *Let B be a tableau branch for a data base instance r with integrity constraints IC, say $B = I \cup r$. B is closed iff one of the following conditions holds:*

1. *$a = b \in B$ for different constants a, b in D.*
2. (a) *$P(\bar{c}) \in I$ and $P(\bar{c}) \notin r$, for a ground tuple $\bar{c}$ containing elements of D only.*
   (b) *$P(\bar{c}) \in I$ for a ground tuple containing (some) Skolem symbols, when there is no way to define them inside the active domain and over the whole tableau in such a way that $P(\bar{c})$ becomes an element of r.*
3. *$\neg P(\bar{c}) \in I$ and $P(\bar{c}) \in r$ for a ground tuple $\bar{c}$ containing elements of D only.*
4. *$\varphi \in B$ and $\neg\varphi \in B$, for an arbitrary formula φ.*
5. *$\neg t = t \;\in B$ for any term t.*

Condition 1. takes into account the unique names assumption. Notice that it is restricted to data base constants, so that it does not consider new parameters. Condition 2(a) takes into account the closed world assumption. If $P(\bar{c}) \in I$, this means that $P(\bar{c})$ belongs to a model of IC. But if $P(\bar{c}) \notin r$, by the closed world assumption, $\neg P(\bar{c})$ holds in r. Hence B is closed. Alternative condition 2(b) (actually it subsumes 2(a)) gives an account of example 3 and the last two cases in example 4.

In condition 3. one might miss a second alternative as in condition 2., something like "$\neg P(\bar{c}) \in I$ for a ground tuple containing Skolem symbols, when there is no way to define them considering elements of $D \setminus Act(r)$ in such a way that $P(\bar{c}) \notin r$". This condition can be never satisfied because we have an infinite data base domain, but a finite active domain. So, it will never apply. This gives an account of example 5. Conditions 3., 4. and 5. are the usual closure conditions, 3. being a special case of 4. which corresponds to the closure of an IC-branch by r.

Now we can state the main properties of tableaux for data base instances and their integrity constraints.

Proposition 1 *For a data base instance r and integrity constraints IC, it holds:*

1. *r is inconsistent wrt to IC iff every branch of the tableau $TP(IC \cup r)$ is closed.*
2. *$TP(IC \cup r)$ is closed iff r is not a model of IC.*

3.3 Opening tableaux

The inconsistency of a data base r wrt IC is characterized by a tableau $TP(IC \cup r)$, which has only closed branches. Every branch B of $TP(IC \cup r)$ is of the form $B = I \cup r$, where I is the IC-part.

In order to obtain a repair of r, we suppress the literals in the branches which are "responsible" for the inconsistencies, even implicit literals corresponding to the CWA. Every branch which can be "opened" in this way will possibly yield a repair. We can only repair inconsistencies due to literals in r. We cannot suppress literals in I because, according to our approach, integrity constraints are rigid, we are not willing to give them up. We only allow changes in the data base instances. We cannot suppress equalities $a = b$ neither built-in predicates such as for example $t \leq t'$. But we suppose that those types of literals are never part of a data base instance.

According to definition 3, we can repair inconsistencies due only to cases 2. and 3. If B is closed because of the CWA (case 2.), it can be opened by adding $P(\bar{c})$ to r. If B is closed because of contradictory literals $\neg P(\bar{c}) \in I$ and $P(\bar{c}) \in r$, then it can be opened by deleting $P(\bar{c})$ from r.

Example 6 *(Example 2 continued) The tableau has 9 closed branches: (we only align the literals within the branches)*

r_1	r_2	r_3
$Supply(C, D_1, I_1)$	$Supply(C, D_1, I_1)$	$Supply(C, D_1, I_1)$
$Supply(D, D_2, I_2)$	$Supply(D, D_2, I_2)$	$Supply(D, D_2, I_2)$
$Class(I_1, T_4)$	$Class(I_1, T_4)$	$Class(I_1, T_4)$
$Class(I_2, T_4)$	$Class(I_2, T_4)$	$Class(I_2, T_4)$
$\neg Supply(C, D_1, I_1)$	$\neg Supply(C, D_1, I_1)$	$\neg Supply(C, D_1, I_1)$
$\neg Supply(D, D_2, I_2)$	$\neg Class(I_2, T_4)$	$D = C$
r_4	r_5	r_6
$Supply(C, D_1, I_1)$	$Supply(C, D_1, I_1)$	$Supply(C, D_1, I_1)$
$Supply(D, D_2, I_2)$	$Supply(D, D_2, I_2)$	$Supply(D, D_2, I_2)$
$Class(I_1, T_4)$	$Class(I_1, T_4)$	$Class(I_1, T_4)$
$Class(I_2, T_4)$	$Class(I_2, T_4)$	$Class(I_2, T_4)$
$\neg Class(I_1, T_4)$	$\neg Class(I_1, T_4)$	$\neg Class(I_1, T_4)$
$\neg Supply(D, D_2, I_2)$	$\neg Class(I_2, T_4)$	$D = C$
r_7	r_8	r_9
$Supply(C, D_1, I_1)$	$Supply(C, D_1, I_1)$	$Supply(C, D_1, I_1)$
$Supply(D, D_2, I_2)$	$Supply(D, D_2, I_2)$	$Supply(D, D_2, I_2)$
$Class(I_1, T_4)$	$Class(I_1, T_4)$	$Class(I_1, T_4)$
$Class(I_2, T_4)$	$Class(I_2, T_4)$	$Class(I_2, T_4)$
$C = C$	$C = C$	$C = C$
$\neg Supply(D, D_2, I_2)$	$\neg Class(I_2, T_4)$	$D = C$

Each branch consists of an I-part and the r-part $r_i = r \cup I_i$. And we have

I_1	I_2	I_3
$\neg Supply(C, D_1, I_1)$	$\neg Supply(C, D_1, I_1)$	$\neg Supply(C, D_1, I_1)$
$\neg Supply(D, D_2, I_2)$	$\neg Class(I_2, T_4)$	$D = C$

I_4	I_5	I_6
$\neg Class(I_1, T_4)$	$\neg Class(I_1, T_4)$	$\neg Class(I_1, T_4)$
$\neg Supply(D, D_2, I_2)$	$\neg Class(I_2, T_4)$	$D = C$

I_7	I_8	I_9
$C = C$	$C = C$	$C = C$
$\neg Supply(D, D_2, I_2)$	$\neg Class(I_2, T_4)$	$D = C$

In order to open this closed tableau, we can suppress literals in the closed branches. Since a tableau is open whenever it has an open branch, each opened branch of the closed tableau might produce one possible transformed open tableau. Since we want to modify the data base r, which should become consistent, we should try to suppress a minimal set of literals in the r-part of the branches in order to open the tableau. This automatically excludes literals like e.g. $D = C$ which are certainly not data base literals, but come from constraints.

In this example we observe that the data base literals of some of the I_j are subsumed by others. Let us denote by I'_j the set of data base literals in I_j. We have then $I'_1 \supset I'_7$, $I'_2 \supset I'_8$, $I'_3 \supset I'_9$, $I'_4 \supset I'_7$, $I'_5 \supset I'_8$, $I'_6 \supset I'_9$. But this shows that in order to open r_1, for example, we have to suppress more literals in r than for opening r_7. Hence, we can decide that the branches whose data base part contains the data base part of another branch can be ignored: they will not produce any (minimal) repairs. This allows us not to consider r_1 through r_6 in our example. Since r_9 cannot be opened by suppressing a data base literal, we need to consider r_7 and r_8 only for obtaining repairs.

The following lemma tells us that we can ignore branches with subsumed I-parts, because those branches cannot become repairs.

Lemma 1 *If $r \subseteq r' \subseteq r''$, then $r' <_r r''$.*

Moreover, as illustrated above, where the tableau tree is shown, sometimes we can detect possible subsuming branches without fully developing the tableau. In example 2 the first formula has been split by a tableau rule and we have already closed two branches. When we apply another rule, we know then, that the branch $C = C$, which is not closed yet, will be not closed or closed by less data base literals than the first two branches.

Let us now define what is tableau opening.

Definition 4 *Let $B = I \cup r$ be a closed branch of the tableau $TP(IC \cup r)$. r' is called* opening *of r wrt B (and $B' = I \cup r'$, an opening of B) iff $r' = A \cup B \cup C$, where A, B, C are defined according to conditions in definition 3, resp., namely:*

(a) A contains all the ground atoms $P(\bar{c})$ satisfying condition 3.2(a).

(b) B contains one data base literal $P(\bar{c}_d)$ for each ground atom $P(\bar{c})$ satisfying condition 3.2(b), where $\bar{c}_d$ is obtained from $\bar{c}$ replacing Skolem terms in $\bar{c}$ by elements in $D \setminus Act(r)$.[4]

(c) C contains all the atoms in r except for those $P(\bar{c})$ satisfying condition 3.3.

[4]This case will appear when we start from existential integrity constraints.

In consequence, for an opening $I \cup r'$ of a branch $I \cup r$ it holds:(a) If $P(\bar{c}) \in I$ and $P(\bar{c}) \notin r$, then $P(\bar{c}) \in r'$. (b) If $\neg P(\bar{c}) \in I$ and $P(\bar{c}) \in r$ then $P(\bar{c}) \notin r'$. Notice that we only open branches which are closed to conflicting data base literals. Every opening is related to a possibly non minimal repair of the original data base instance. For repairs, we are only interested in "minimally" opened branches, i.e. in open branches which are as close as possible to r. Openings of r are obtained by deletion or insertion of atoms from/into r. In order to obtain minimal repairs, we have to make a minimal set of changes, therefore we do not keep openings associated to an r'', such that $r' \Delta r \subsetneq r'' \Delta r$, for some other r' associated to another opening. Now we need to compare openings corresponding to different closed branches in the tableau. We want the minimal ones.

Definition 5 *Let r' and r'' be openings of r in $TP(IC \cup r)$; $r' = (r \setminus r_1') \cup r_2'$ and $r'' = (r \setminus r_1'') \cup r_2''$. r' is closer to r than r'' iff $r_1' \subseteq r_1''$ and $r_2' \subseteq r_2''$. This is denoted by $r' \prec_r r''$. r' is a* minimal *opening of r in IC iff r' is a minimal element of the set of opened branches of $TP(IC \cup r)$ wrt. the "ordering" relation $\prec_r$.*

Now, it can be shown that minimal openings correspond to data base repairs.

Theorem 1 *Let r be an inconsistent data base wrt IC. Then r' is a repair of r iff $I \cup r'$ is a minimal opening of a closed branch $I \cup r$ in $TP(IC \cup r)$.*

Example 7 *(Example 2 continued) $TP(IC \cup r)$ has two minimal openings:*

r_1	r_2
$S(C, D_1, I_1)$	$S(C, D_1, I_1)$
$Class(I_1, T_4)$	$S(D, D_2, I_2)$
$Class(I_2, T_4)$	$Class(I_1, T_4)$

The last closed branch cannot be opened because it is closed by the atom $D = C$ which is not a data base predicate.

4 Consistent Query Answering

In order to determine consistent answers to queries, we can also use, at least at the theoretical level, a tableaux theorem prover that is able to produce $TP(IC \cup r)$ and its openings. Let us denote by $op(TP(IC \cup r)$ the tableaux $TP(IC \cup r)$ with its minimal openings, that is with its branches that lead to minimal openings, open.

We claim that, according to definition 2, $\bar{c}$ is a consistent answer to the open query $Q(\bar{x})$ when $op(TP(IC \cup r)) \otimes TP(\neg Q(\bar{c}))$ is, again, a closed tableaux, where $T \otimes T' = \{X \cup Y : X \in T \text{ and } Y \in T'\}$, is the combined tableau. In consequence, we might use the tableaux $op(TP(IC \cup r)) \otimes TP(\neg Q(\bar{x}))$ in order to retrieve those values for $\bar{x}$ that restore the closure of all the opened branches in the tableaux.

Example 8 *Consider the following integrity constraints IC: $\forall(x, y, z, u, v)$ $(Student(x, y, z) \wedge Student(x, u, v) \rightarrow y = u \wedge z = v)$; and the inconsistent students data base instance*

$$
\begin{array}{c}
r \\
Student(S_1, N_1, D_1) \\
Student(S_1, N_2, D_1) \\
Course(S_1, C_1, G_1) \\
Course(S_1, C_2, G_2)
\end{array}
$$

which has the two repairs

$$
\begin{array}{cc}
r_1 & r_2 \\
Student(S_1, N_1, D_1) & Student(S_1, N_2, D_1) \\
Course(S_1, C_1, G_1) & Course(S_1, C_1, G_1) \\
Course(S_1, C_2, G_2) & Course(S_1, C_2, G_2)
\end{array}
$$

Let $Q(y) : \exists z Course(S_1, y, z)$ *be an open existential query. The tableaux for* $op(TP(IC \cup r)) \otimes TP(\neg \exists z Course(S_1, y, z))$ *is closed, actually by* $y = C_1$ *and* $y = C_2$.

5 Conclusion

The methodology for answering queries deserves further investigation. In particular, it would be interesting to compare it with the methodology presented in [3] for consistent query answering, that relies on denials and disjunctive forms for constraints. As shows the example 8, one advantage of the method proposed here is that we can answer to existential queries, which is not possible with the Celle and Bertossi's approach.

It would be also interesting:

1. To develop a methodology for consistently query answering that does not produce all the repairs, but just processes the opened (or maybe the original) tableau.

2. Alternatively, to obtain from the tableau the right "residues" in order to modify (rewrite) the original query [1].

3. To take advantage of a DBMS in order not to use explicitly in the tableau generation process the explicit DB instance.

4. To use a theorem prover (in combination with a DBMS) in order to consistently answer queries.

References

[1] M. Arenas, L. Bertossi and J. Chomicki. Consistent Query Answers in Inconsistent data bases. Proc. ACM Symposium on Principles of data base Systems (ACM PODS'99), 1999, pp. 68–79.

[2] E. W. Beth. *The Foundations of Mathematics.* North Holland, 1959.

[3] A. Celle and L. Bertossi. Querying Inconsistent data bases: Algorithms and Implementation. In 'Computational Logic - CL 2000', J. Lloyd et al. (eds.). Stream: 6th International Conference on Rules and Objects in data bases (DOOD'2000). Lecture Notes in Artificial Intelligence 1861, Springer 2000, pp. 942 – 956.

[4] M. Fitting. First Order Modal Tableaux. Journal of Automated Reasoning, 4(2), 191–213, 1988.

[5] M. Fitting. *First Order Logic and Automated Theorem Proving.* Texts and Monographs in Computer Science. Springer Verlag, 1990.

[6] E. Lafon and C. B. Schwind. A Theorem Prover for Action Performance. In Y. Kodratoff, editor, *Proceedings of the 8th European Conference on Artificial Intelligence*, pages 541–546. Pitman Publishing, 1988.

[7] R. Reiter. Towards a Logical Reconstruction of Relational data base Theory. In "On Conceptual Modeling", Brodie, M. L. and Mylopoulos, J. and Schmidt, J. W. (eds.), Springer-Verlag, 1984, pp. 191–233.

[8] C. B. Schwind. A Tableau-based Theorem Prover for a Decidable Subset of Default Logic. In M. E. Stickel, editor, *Proceedings of the 10th International Conference on Automated Deduction*, number 449 in Lecture Notes in Artificial Intelligence, pages 541–546. Springer Verlag, July 1990.

[9] C. B. Schwind and V. Risch. Tableau-based Characterisation and Theorem Proving for Default Logic. *Journal of Automated Reasoning*, 13(4):223–242, 1994.

[10] R. M. Smullyan. *First Order Logic.* Springer, 1968.

[11] M. Winslett. Reasoning about Action with a Possible Models Approach. In *Proceedings of the 8th National Conference on Artificial Intelligence*, pages 89–93, 1988.

A Proofs, Intermediate Results and Further Examples

It is possible to prove [1] that for every data base instance r and set IC of integrity constraints, there is always a repair r'. If r is already consistent, then r is the only repair.

The following lemma will be useful.

Lemma 2

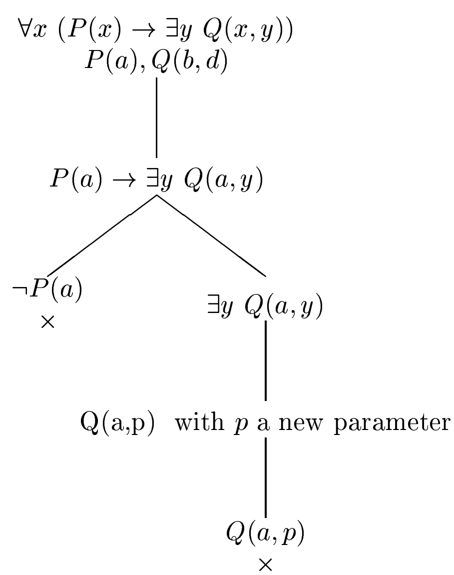
1. *If $r' \leq_r r''$, then if $l \in r \cap r''$, then $l \notin r'$.*
2. *If $r' \subseteq r$, then $r \Delta r' = r \setminus r'$.*

Proof:

1. *Let be $r \Delta r' \subseteq r \Delta r''$; from this follows that $r \cap r'' \subseteq r \cap r'$, from which the lemma follows.*
2. *straightforward*

Q.E.D.

Example 9 *Consider the referential IC : $\forall x\ (P(x) \rightarrow \exists y\ Q(x,y))$, and the inconsistent data base instance $r = \{P(a), Q(b,d)\}$, for $a, b, c \in D$. We can develop the following tableau $TP(IC \cup r)$:*



The branch on the right-hand side is closed because $Q(a,p) \notin r$. As in the previous example, we did not develop the whole tree. For example, we can successively apply the γ-rule to formula $\forall x\ (P(x) \rightarrow \exists y\ Q(x,y))$ replacing x by b, d, etc. But since the tree is closed, we do not develop it any further.

Example 10 *Consider a modification of the referential IC we had in example 9. Now $IC:\ \forall x\ (P(x) \rightarrow \exists y\ (Q(x,y) \wedge \forall z\ (Q(x,z) \rightarrow y = z)))$, and the inconsistent data base instance $r = \{P(a), Q(a,e), Q(a,f)\}$. We have the following tableau for this combination:*

$$\forall x\ (P(x) \rightarrow \exists y\ (Q(x,y) \wedge \forall z\ (Q(x,z) \rightarrow y = z)))$$
$$P(a), Q(a,e), Q(a,f)$$

$\neg P(x)$
$\times$ with x = a

$Q(a,p)$ with p a new parameter
$\forall z\ (Q(a,z) \rightarrow p = z)$

$\neg Q(a,z)$
$\times$ with $z = e$

$p = z$
$p = e$ with $z = e$
$p = f$ with $z = f$

$e = f \quad (*)$
$\times$

Step $(*)$ can be obtained applying transitivity to the two preceding formulas. Then, the branch can be closed by condition 1. in definition 3.

Lemma 3 *Let φ and ψ be any formulae.*

1. $TP(\{\varphi \vee \psi\}) = TP(\{\varphi\}) \cup TP(\{\psi\})$
2. $TP(\{\varphi \wedge \psi\}) = \{X \cup Y : X \in TP(\{\varphi\})$ *and* $Y \in TP(\{\psi\})\}$
3. *If* $B \in TP(\varphi \wedge \psi)$ *then* $B = B' \cup B''$ *and* $B' \in TP(\varphi)$ *and* $B'' \in TP(\psi)$.

Property 3, 3 follows directly from property 3, 2.
These properties gives rise to the following notation.

$$T \otimes T' = \{X \cup Y : X \in T \text{ and } Y \in T'\}$$

These properties can be used to check whether a formula φ derives from a theory A. $A \models \varphi$ iff $A \rightarrow \varphi$ is a theorem, what will be proved if we derive a contradiction from assuming $\neg(A \rightarrow \varphi)$. Therefore we will have to compute $TP(\{\neg(A \rightarrow \varphi)\})$ and check for closure. Using the second property, we will check $TP(\{A\}) \otimes TP(\{\neg\varphi\})$ for closure, allowing us to compute $TP(A)$ only once for any number of requests.

Proof: of Lemma 1: Immediately by $(I \cup r)\Delta r = (I \setminus r)$ Q.E.D.